489668

30

RADC-TR-66-474, Volume III
Final Rep

RELIABILITY CENTRAL AUTOMATIC DATA PROCESSING SUBSYSTEM

Data Management System Survey

Auerbach Corporation

DDC FILE COPY

TECHNICAL REPORT NO. RADC-TR-66-474
August 1966

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of RADC (EMLI), GAFB, N.Y. 13440.

Rome Air Development Center
Research and Technology Division
Air Force Systems Command
Griffiss Air Force Base, New York

ACCESSION for

CFSTI	WHITE SECTION ☐
DDC	BUFF SECTION ✓
UNANNOUNCED	
JUSTIFICATION	w Statement 2 Ltr

BY [illegible]

DISTRIBUTION/AVAILABILITY CODES

DIST.	AVAIL. and/or SPECIAL	
2		

⑱ RADC

⑲ TR-66-474-VoL-3

⑥

RELIABILITY CENTRAL AUTOMATIC DATA PROCESSING SUBSYSTEM.

Volume III. Data Management System Survey,

⑨ Final rept.,

Auerbach Corporation

⑮ AF30(602)-3820

⑯ AF-5519

⑩ J. Sable, W. Crowley, M. Rosenthal S. Forst
P. Harper

This document is subject to special
export controls and each transmittal
to foreign governments or foreign
nationals may be made only with
prior approval of RADC (EMLI),
GAFB, N.Y. 13440.

⑪ Aug 66

⑫ 50p. ⑭ 1280-TR-VoL-3

MAINLINE ENGINEERING MAPLE SHADE, N.J.
9/23/66 240

1473
[illegible]

047 050

FOREWORD

This three-volume final technical report was prepared by the Auerbach Corporation, Philadelphia 3, Pennsylvania under Contract AF 30(602)-3620, Project 5519. It is identified by the contractor as 1280-TR. The authors were Dr. J. Sable, W. Crowley, M. Rosenthal, S. Forst, and P. Harper. The Rome Air Development Center Project Engineer was Casper DeFiore, EMIID.

This technical report contains information embargoed from release to the Clearinghouse for Federal Scientific and Technical Information, Department of Commerce, by AFR 400-10.

This technical report has been reviewed and is approved.

Approved: [illegible]
FRANK J. TOMAINI
Chief, Information Processing Branch

Approved: [illegible]
JAMES J. DIMEL, Colonel, USAF
Chief, Intel & Info Processing Division

FOR THE COMMANDER: [illegible]
IRVING J. GABELMAN
Chief, Advanced Studies Group

PAGES NOT FILMED ARE BLANK

ABSTRACT

This report is a continuation and updating of a previous survey of data management systems (RADC-TR-189, Vol. II, July 1965 (AD-469-269)).

GIS (IBM), IDS (GE), and MULTICS (BTL GE, MIT) are discussed and compared with DM-1 (AUERBACH Corporation). Of these systems, only DM-1 (described in Volumes I and II of this Final Report) and GIS have all the characteristics of a generalized data management system, as the term is defined in this report. IDS is a programming system with extended random-access capability, and MULTICS is an operating system for a multiuser, multiprocessing system.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I. INTRODUCTION

This comparative survey of large-scale data management systems is submitted to Rome Air Development Center in accordance with Change "A" to Contract AF 30 (602) - 3820, dated 14 October 1965, and is part of the Final Report for that contract (Completion of ADPS DESIGN for Reliability Central). This report also represents a continuation and updating of a survey of data management systems submitted to RADC as part of a prior contract.*

These surveys are intended to document current techniques in data management, and to indicate how the AUERBACH data management system for the Reliability Analysis Central, DM-1, ** has drawn on available data management technology, and where it has contributed to that technology.

* J. Sable, et al: Design of Reliability Central Data Management Subsystem. RADC-TR-65-189, Vol II., July 1965 (AD-469-269), Contract AF 30 (602) - 3433.

** DM-1 is AUERBACH Corporation's designation for the data management system design referred to in the 1965 report as the Reliability Central Data Management System. References to DM-1 in this volume refer to the system design as documented in Volumes I and II of the 1966 Final Report.

The systems covered in the earlier (1965) survey are as follows:

RECOL (RCA)
ACSIMATIC (RCA)
ASO SYSTEM (University of Pennsylvania)
IPS (IBM)
HQ, USAF SYSTEM (IBM)
COLINGO (MITRE)
LUCID (SDC)
ADAM (MITRE)

In this report, some material on MULTICS (BTL, GE, MIT) is presented, and IDS (GE) and GIS (IBM) are covered in some detail. Of these three systems, only GIS has all the characteristics of a generalized data management system (as the term is defined in Section II). IDS is a programming system, and MULTICS is an operating system.

As stated in the 1965 survey, it is difficult to present a comparative survey of several systems in one technical area, without giving the erroneous impression that judgment is being passed on the approach taken in the systems under consideration. There is no such intent in this report. In order to make comparative analysis meaningful, the functions, design features, and operational capabilities of the AUERBACH DM-1 are compared with each of the other systems. These comparisons should be treated as comparisons of facts, not as judgments as to whether the features planned for DM-1 should have been present in the other systems or vice versa.

For the benefit of the reader who has not read Volumes I and II of this report, DM-1 is an extended data management and operating system for data processing jobs. It provides a user-oriented command language and query system, but its programmer interface is via a service package accessible through a procedure call (subroutine mode). It does not contain a unique procedural language, but it will accept (as does any operating system) programs compiled from any source language. However, all references to the data base are made as logical item requests via service package calls so that no physical I/O statements exist in the source program (therefore, no I/O instructions appear in the object program). This allows the programmer to code in any appropriate procedural language and allows data references to be parameterized and bound for the service package at job execution time.

A set of system support jobs exists in DM-1. These jobs, which can be requested via a user-oriented command language, are used to define new data structures for the data base, incorporate and maintain the data, and enter new program and job definitions.

SECTION II. TERMINOLOGY

Many data processing centers exist in a problem environment which is characterized by a large data base and stringent requirements for the rapid and accurate assimilation of new data, changes in the data structure, interrogation of data content, and analytic results based on data content. Examples of such problem environments include intelligence and command and control applications in the military and management information and inventory control applications in industry, to name a few. A data management system is defined as the software environment which the data processing center provides as a tool to a staff of users, programmers, and operators responsible for maintenance, query, and analysis of the data base.

In order to sharply focus the discussion of system characteristics in this study, it will be important to discriminate among different, albeit closely related, concepts. Because of the still early stage of development of the data management system field, and the utilization of equally new techniques in time and facility sharing, these concepts have not had the benefit of standardization of terminology.

One aspect that is essential to the clear understanding of data management systems is the distinction between data content and data structure. Data structure refers to the relationship between items (as named classes of data) such as the generic relationship among EMPLOYEE NAME, EMPLOYEE RECORD, and PERSONNEL FILE in a corporate data base. The data content of a data management system is the set of data values that comprise declarative statements concerning the objects of the environment outside the machine system which are of interest to the users of the system (such as the names of specific employees). This collection of declarative data about the outside world is called the data base and in some sense, represents an abstract model of the objects in the outside world. The data base makes up what is usually the majority (but by no means all) of the data in the data management system. A large part of the data in the system may pertain to the definition of data (names, structure, and format of data items) and its location in the data base. This data-descriptive data (data about data) is located in the system directory (or directories).*

It is important to distinguish clearly between the physical and logical aspects of data structure. Physical structure refers to the disposition of data on the storage media and is usually connotated by such words as: bit, byte, word, page, segment, block, track, reel, etc. These words refer to the physically accessible data elements that are recognizable by the hardware at various processor and accessibility levels. In particular, a page is the allocatable unit in core storage. Each page occupies a known block of contiguous locations in core. A block, sometimes called segment,* is an identifiable program or data unit which can occupy a page and can exist both in core and in external storage media, possibly in more than one copy.

Logical structure refers to the named data elements and their relationships. The named data elements are called items and can be categorized into compound (structured) items and simple (elementary and/or terminal) items. Compound items are items

* One of the systems discussed later, IDS, does not have separate directories, but carries structural information with the data and compiled with its programs.

** The word segment is used in a special sense by the designers of GIS. It's local definition appears in Paragraph 3.1 in that system report.

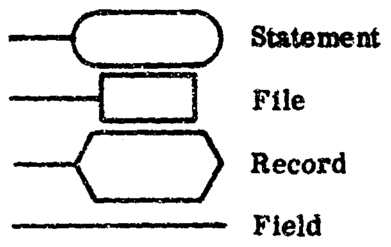
made up of other items; that is, they have a nested structure. Simple items are variables or names of properties or attributes of outside-world elements. They take data values; that is, they are unstructured fields in the data base which can be characterized as to type (binary, integer, alphanumeric, etc.) and length (fixed length or variable length). For example, MANUFACTURER may be the name of a simple item or field in the data base which, in a particular instance, has the value RCA. Synonymously, one may say the attribute MANUFACTURER has the value RCA. The outside-world elements that are described by the declarative data of the data base are messages, events, and/or objects which form the reason-for-being of the system and are the main concern of its users.

The term data base itself can be thought of as the name of the maximum item of data — the root of a data structure tree and all its subnames. The data base is composed of lesser compound items called statements, files, records, and links. A file is a compound item made up of an arbitrary number of items called records which have the same logical structure. That is, a replicated item and its parent are called a record and a file, respectively. A link is an item that refers to another item which is physically remote. There are two logical interpretations of the relationship between the source item containing the link and the target item to which it points. One is a generic-specific relationship in which the target item is logically embedded (but physically non-embedded) in the source item (and therefore an item with more than one parent). Another interpretation is an ordering relationship in which the source item precedes the target item (with no generic relationship implied). The correct interpretation may be implicit in the processor, determined by context or position, or may be indicated by an explicit "tag."

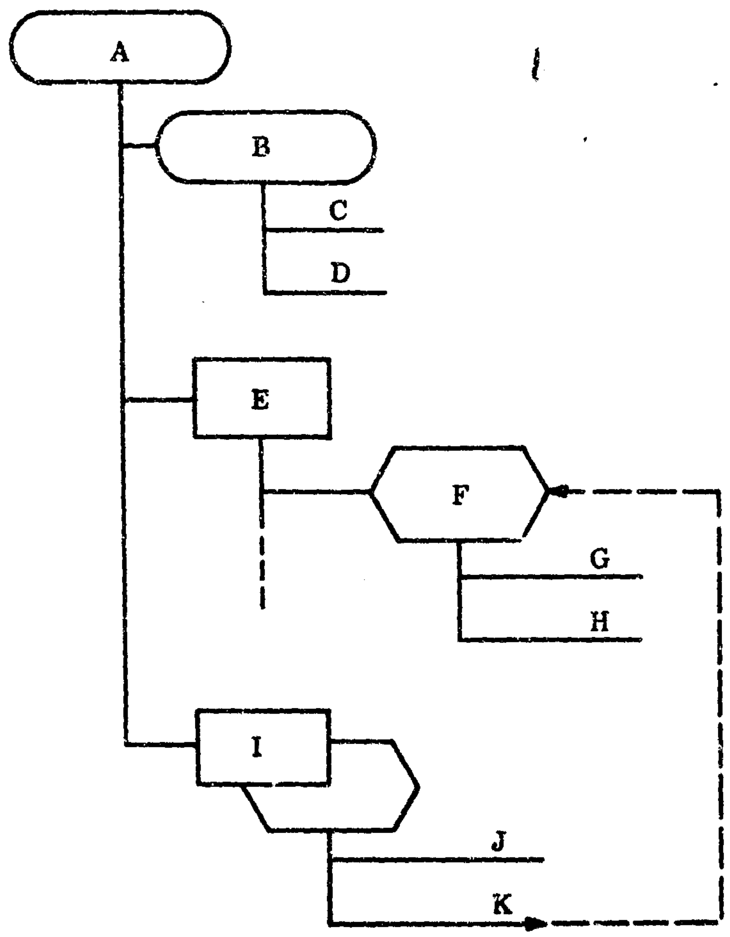
In some systems, the term repeated group or repeated set is used to refer to files that occur within records. The number of nesting levels of items within items is logically unlimited and may reflect the hierarchic structure of objects in the outside world (e.g., assemblies and sub-assemblies of components in equipment and systems).

The term statement is used to denote a compound item made up of any fixed structure. Thus files or records are statements of a special kind.

The following symbols will be used in illustrating logical data structures in this survey:



The following hypothetical structure employs these symbols:



In this structure, statement A is made up of three items: statement B and files E and I. Statement B is made up of simple items, or fields, C and D. Note the alternate representation for records of files E and I. The representation used for I records will be used when the record name is the same as the file name. The I records contain two fields, J and K. Field K is indicated as being a link which points to one or more F records.

SECTION III. SYSTEM REPORTS

3.1 GIS SYSTEM REPORT

3.1.1 Objectives

GIS (Generalized Information System) is a comprehensive Data Management System that is being developed by IBM Federal Systems Division. It is designed to operate within Operating System/360 to support a wide variety of data processing applications by providing facilities for defining, maintaining, and processing data files and unformatted text.

GIS provides a user-oriented interface which permits the description of logical data relationships and processing tasks without concern for the processing devices, physical formats, and device operations involved in manipulating the data.

A number of variable parameter statements, which together constitute the GIS language, are available to allow a user to establish and modify the more static control tables, to enter task specifications, and to control task execution. The parameter content of the control tables and task specification are combined at execution time to direct and control the flow of program operations. A wide range of file structures and organizations can be established, maintained, and used.

3.1.2 Implementation and Status

GIS is a set of programs for System/360 that are designed to perform data file establishment, maintenance, retrieval, and presentation operations that are common to many data processing applications. GIS is expected to function on a range of System/360 configurations under control of OS/360. It may be operated in either a job sequential mode or a teleprocessing (multiprogramming) mode with remote terminals. The minimum configuration for the job sequential mode is Model 30 F (65,536 bytes of core storage) and, for the teleprocessing mode, a Model 40G (131,072 bytes of core storage). In addition, GIS programs require approximately 500,000 bytes of direct access secondary storage and an interval timer.

GIS programs are run as jobs under the OS/360. The implementation language for the GIS translator and system programs is Assembler Language. Parts of GIS are expected to be available in 1967.

Although quite different in concept and design from its predecessors, GIS is the latest in an evolutionary line of IBM file control systems, represented by TUFF, TUFF/TUG, FFS, and IPS, * developed for military operations control centers by IBM Federal Systems Division. GIS will be supported by an IBM Applications Group (Type II support).

3.1.3 System Description

3.1.3.1 User Languages

3.1.3.1.1 General. Although GIS runs as a job within OS/360, it has many of the features of an operating system itself. A GIS job is made up of a series of JOB control cards followed by a task specification. Within each task GIS will exercise control over a number of GIS and non-GIS (user supplied) programs.

OS/360 provides the overall control at a given installation concerned with receiving, queueing, and executing jobs. Core memory allocation, channel time allocation, program loading, and multiprogramming functions are all handled by OS/360. The GIS

* See RADC-TR-65-189, Vol. II for a discussion of IPS.

run is initiated by OS/360 in response to the GIS JOB cards. GIS itself has a control mechanism which interprets the task specification.

The task specification contains task control information, the names of generalized GIS service functions (e.g., CALL and QUERY), the parameters needed to perform the desired action, and user-supplied (non-GIS) programs.

There are two basic modes of GIS operation: JOB-oriented and TP-oriented (telecommunication processing-orientation). The TP-oriented mode takes advantage of the capability of OS/360 to handle multiple messages from the telecommunication devices attached to the system. In the TP-oriented mode, multiprogramming controls in OS/360 make it unnecessary to initiate a new job for each new task specification. In the JOB-oriented mode of operation, the user may enter the job from a variety of devices, including telecommunication devices, but, as stated above, must initiate a new job for each task specification. The job is then processed in accordance with the priority assigned by the originator.

3.1.3.1.2 Job Specification Language. GIS provides a language for specifying the tasks that compose a job, their parameters, and the condition for their execution. Task specifications can be saved by GIS and executed when called by the user.

The format for the request to save a task specification is given below:

SAVE INPUT STATEMENT SET <task name> <task specification>

A task specification in the GIS Language is a series of commands and conditional statements to execute GIS and user-supplied (non-GIS) programs. A task specification may contain open (formal) parameters whose values must be supplied by the task specification which names them, or by the user at request time.

A GIS program named CALL will bind the formal parameters of a program. Its parameters are a task name and its parameter value list:

CALL <task name> <parameter list>

The parameter list must supply values for all the formal parameters of the task named. Formal parameters in the task specification are named

$ 1 $, $ 2 $,...

indicating the sequence in which their values must be supplied.

Operation of this program or any other GIS program in a job requires a command of the form:

```
OPERATE GIS PG <program name>
PARAMETER EQ <parameter list>
```

For example, the statement:

```
OPERATE GIS PG CALL PAYROLL PARAMETER EQ AUG 7
```

will cause the GIS program "CALL" to bind the user supplied program "PAYROLL" with its parameter "DATE" = AUG 7

For user (non-GIS) programs in which no parameter binding is necessary, the command is of the form

```
OPERATE <program name>
```

3.1.3.1.3 Query Language. As part of the job specification and request language discussed in the preceding paragraph, GIS has a condition-defining capability which permits a logical (Boolean) condition that must be satisfied by each record before it is eligible for further processing. Conditional statements can be inserted in any task specification as a condition for further processing, such as retrieval, data reduction, updating, reporting, etc. When combined with a retrieval, reporting, or tabulating function, GIS accomplishes what is called a QUERY. In a query, certain functional and data reduction operations to be carried out, as well as the format in which the data is to be presented, can be specified.

A QUERY task specification begins with the word QUERY, followed by a file name, a logical condition to be satisfied by records of the file, and a statement specifying the data to be extracted and the processing to be carried out on eligible records. Because of the file name requirement, a QUERY is limited to a single, identifiable file.

3.1.3.1.3.1 Conditional Statements. The conditional statements involve terms which take the following form:

```
<name of field> <comparison to be made> <basis of comparison>
```

These terms may be combined with AND, OR, and AND NOT operators. Parentheses may be used as required. For example,

IF AGE EQ 21 AND HAIR EQ BLOND OR BROWN

<name of field> may be any valid field name in the file being addressed.

<basis of comparison> may be any field value.

<comparison to be made> may include any of the following operators:

$\{=, >, <, \neq, \geq, \leq,$ BETWEEN, SCAN, MASK, CHANGE$\}$

The SCAN operator tests the field for the particular pattern of characters specified. The MASK operator permits "don't-care" positions in the scan. The CHANGE operator does not require a basis of comparison. It will indicate "true" for any value which is different from the value of the same field in the previous record.

The truth value of each term is evaluated with respect to the appropriate data elements. These truth values are then combined according to the condition (IF — statement), following the usual rules of Boolean algebra with respect to the AND, OR, and AND NOT connectives.

Those data items for which the condition is true are subject to the processing given in the remainder of the QUERY task specification, as indicated below.

3.1.3.1.3.2 Summary Operations. A facility exists to create summaries subsequent to retrieval but prior to final output (i.e., in a QUERY). The following operators are included:

TOTAL — causes the summation of the non-blank entries of the specified field for all selected records.

COUNT — counts the number of non-blank entries of a specified field.

TALLY — counts the number of times a procedure encounters the TALLY statement.

AVERAGE — causes an average to be calculated for a specified field.

3.1.3.1.3.3 Data Presentation. Two commands may be used to obtain data presentations during an update or retrieval task.

LIST – distributes across the page the data (under associated field headings) from the selected fields. A standard format is used and the output appears on the device associated with the query input device.

FORMAT – permits interrupted field headings and data lines, variable width, height, and spacing, and output device selection.

Either command can be pre-stored in the task specification for the query or added at run time.

3.1.3.1.4 Data Description Language. GIS gives the user (e.g., the programmer) the ability to define a file once and then refer to the file and its records and fields, by name (in task specifications), without further reference to its storage format or organization. GIS stores and refers to the data description so that tasks can be executed which reference data elements symbolically. Data organization or format may be changed without necessarily invalidating established task specifications which refer to the modified data descriptions. The data descriptions and file processing controls are stored in a standardized table structure for each file.

3.1.3.2 Data Organization and Structures

3.1.3.2.1 Data Base Organization. The most generic data element in GIS is the file, and each data description must begin with a file name and is, in effect, a file description. The types of structures which can exist in a file, however, are quite varied and general.

A file is composed of an arbitrary number of items, called records, of the same logical structure. A record may have a multilevel structure which may be described as follows: a GIS record is composed of a single "segment"* followed by some defined number of embedded files, where a segment is a string of a defined number of fields (at the same level in the record).

This defines (recursively) what is called a file in GIS. The general logical structure of a GIS file is illustrated in Figure 3-1.

* The word segment is GIS terminology. It is a DM-1 statement whose subitems are fields.

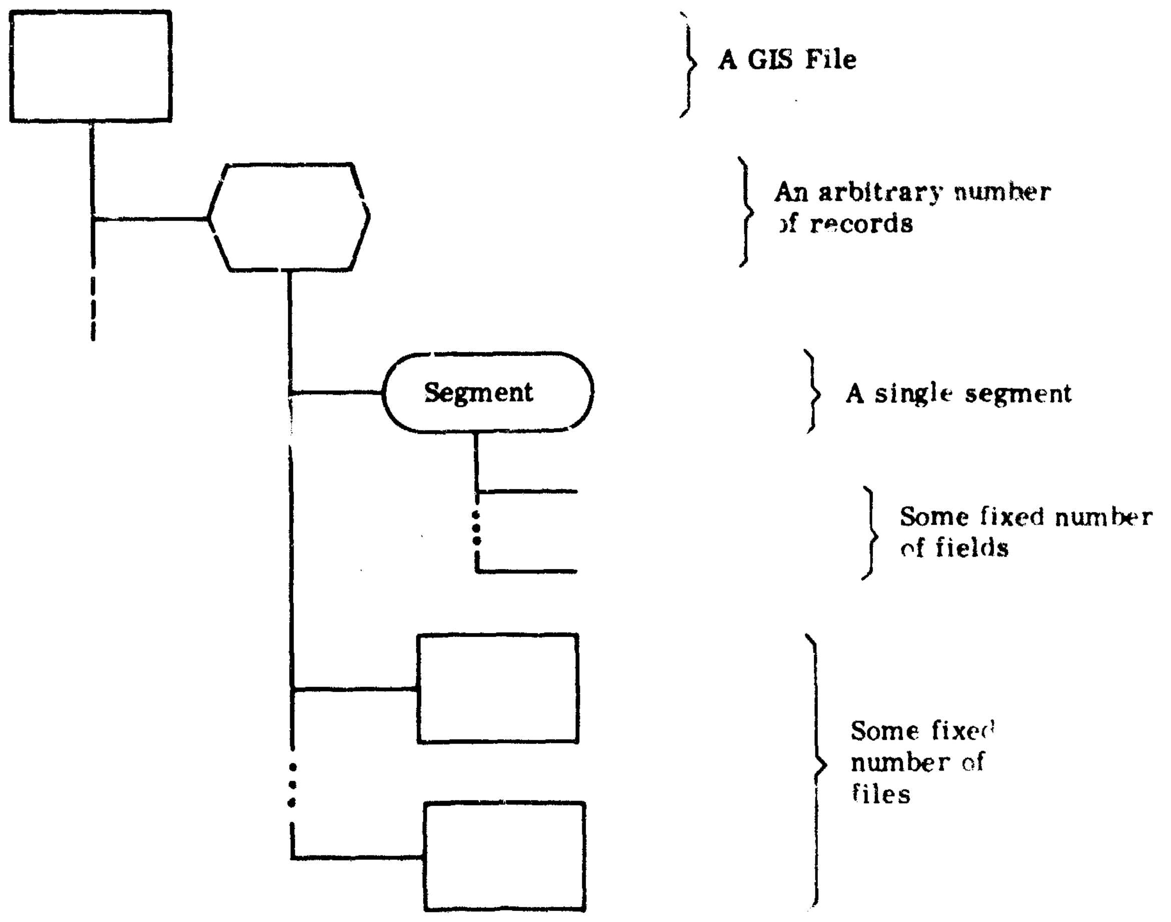

A GIS File
An arbitrary number of records
Segment
A single segment
Some fixed number of fields
Some fixed number of files

Figure 3-1. A GIS File

When there is more than one embedded file in a record (at the same level), each embedded segment must be identified by one of several options (to be described below) such as a key field which is common to all segments at that level and which takes a unique value for each embedded file. An example of a GIS file organization is shown in Figure 3-2, which shows the structure of a hypothetical file called File Zero. The records of File Zero contain a Level 00 segment (made up of fields C, D, and E) and two subfiles. These embedded files, Subfile 1A and Subfile 1B, contain Level 01 segments whose first field is a key field whose value is "A" for Subfile 1A and "B" for Subfile 1B. The Level 01 segment in 1A records contain the additional fields F and G. The Level 01 segment of 1B records contains the field J in addition to the key. The 1A records also contain a Level 02 File. The Level 02 segment need not contain a key field as there is only one Level 02 segment type in the structure. The Level 02 segment contains the information fields H and I.

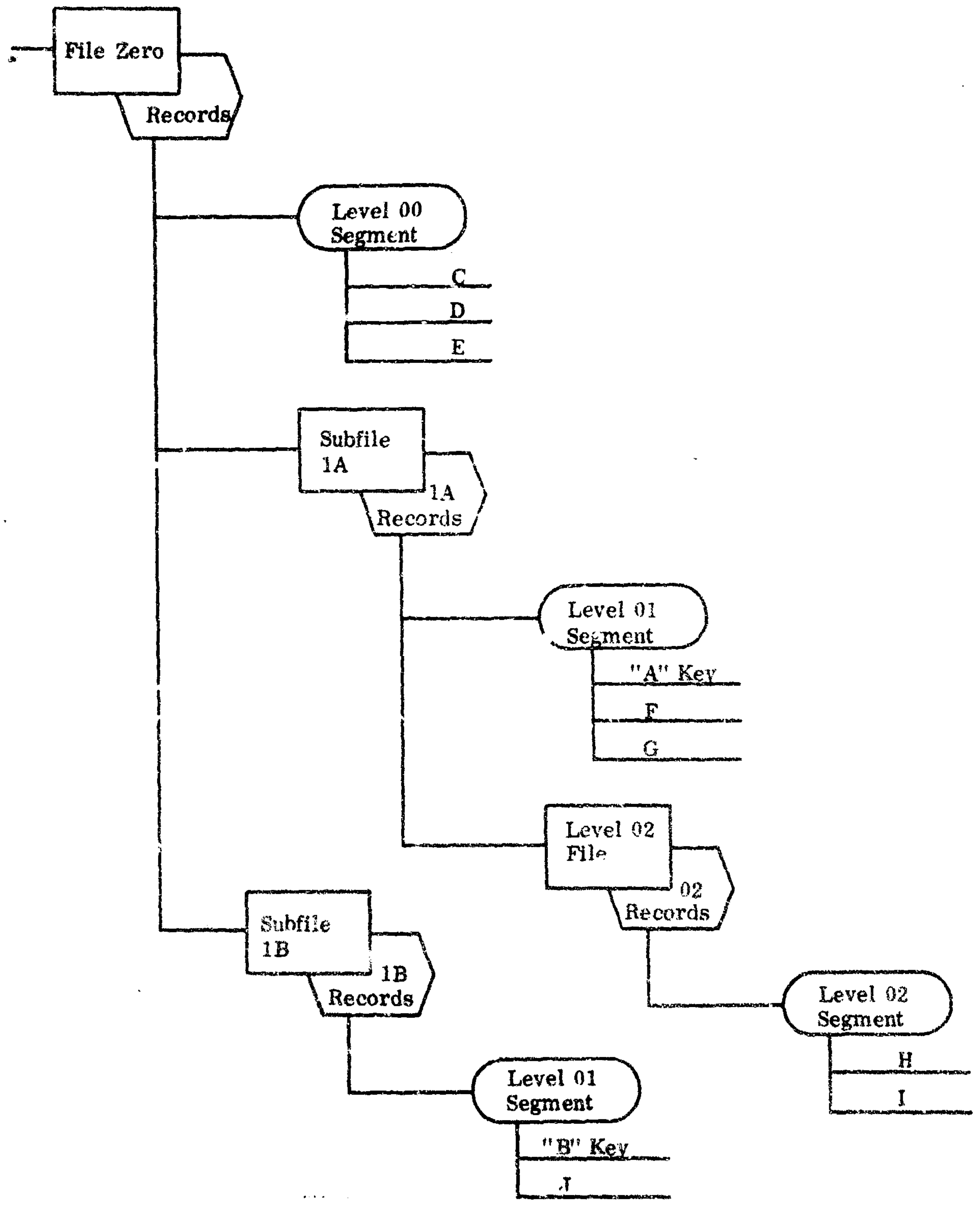


Figure 3-2. A GIS File Example

Since the size of a subordinate file may be different in each parent record, a definition of file size (subrecord occurrence control) must be established for each file. Three options in subrecord occurrence control are provided by GIS, as illustrated in Figure 3-3. In the first option, Figure 3-3(a), each segment must contain a record-count field for each subordinate file (File 1A and File 1B).

In the second option, a key field which is present in every segment must be defined. In Figure 3-3(b), this key is shown as the first field of every segment. Its value is "Z" in every File Zero record, "1A" in every File 1A record, and "1B" in every File 1B record. Thus, the key field serves as an identifier of every segment, thus fulfilling the needs of subrecord occurrence control at the expense of the space for this field in every segment.

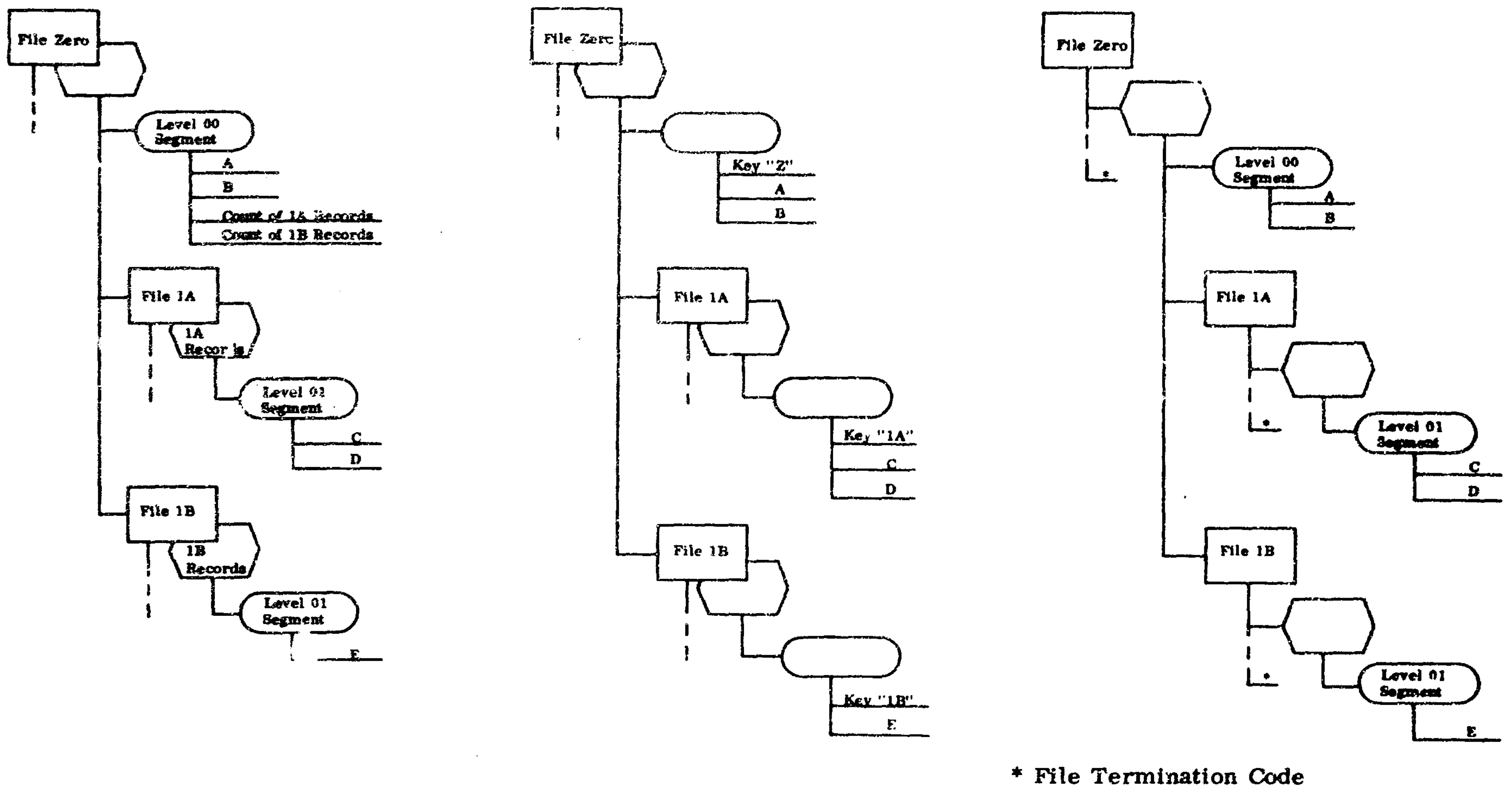
In the third option, a unique one-character termination code is used to signal the end of every file. As indicated in Figure 3-3(c), the File Termination Code occurs as a single character after the last record of every file. While this device is efficient in space, it is usable only where it can be guaranteed that the bit pattern representing the File Termination Code cannot occur as the first character of a segment.*

A field is a terminal GIS item — the smallest unit of data that can be referenced or described. The required descriptive elements for a field are its relative position in the segment, its name (and synonym, if any), its type, and its length. Field type may be any of the following:

(1) Binary,
(2) Decimal,
(3) Floating point,
(4) Alphanumeric.

An alphanumeric field may be either fixed or variable in length, whereas all others must be fixed-length. Length control for a variable-length field may be provided by one of two options: count or terminating code. In the count option, each occurrence of the variable-length field will be preceded by a "length" field indicating the number of characters for this occurrence of the field. In the terminating-code option a user-selected special character is employed to mark the end of each occurrence of the variable-length field.

* The GIS Application Manual implies that this character may not occur in any field — a considerably stronger restriction.



a) Option 1: Count

b) Option 2: Key ıd

c) Option 3: Terminating Code

Figure 3-3. Subrecord Occurrence Control

3.1.3.2.2 Physical Data Formats. In order to accommodate various processing requirements, GIS provides three physical format options for files shown in Figure 3-4. In the first option (linear record), shown in Figure 3-4(a), the data in each record is physically stored in the same sequence as its logical definition. That is, referring to the file shown in Figure 3-3, the fields of the Level 00 segment are followed by all Level 01 segments of File 1A, followed by all Level 01 records of File 1B. (If the 1A records had contained a Level 02 File, those segments would have preceded the Level 01 segments of File 1B.)

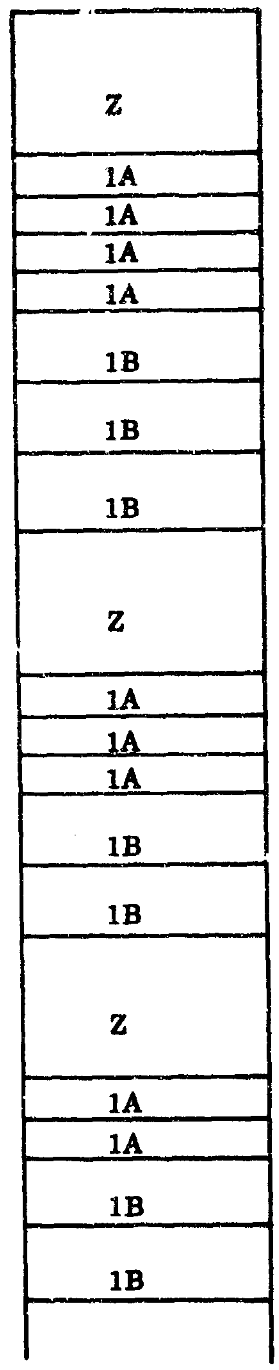
In the second and third options (split records), only the segments in the same level of each file are stored contiguously. In option two (link type split record), shown in Figure 3-4(b), and Figure 3-5(a), a link field occurs as the last field of each embedded segment. This field points from each embedded segment to the location of the segment to which it is subordinate (detail-to-master linkage).

In option three (chain-type split records), shown in Figure 3-4(c) and Figure 3-5(b), a chain of links exists which originates in each master segment and threads through each record of an embedded file and back to the master. Thus, a master segment would contain a link field for each of its embedded files and, also, a link field to the next record (or its master, if it is the last record) at its own level. (Thus, a chain-type split record has the same structure as an IDS record, which is to be discussed in Paragraph 3.2).

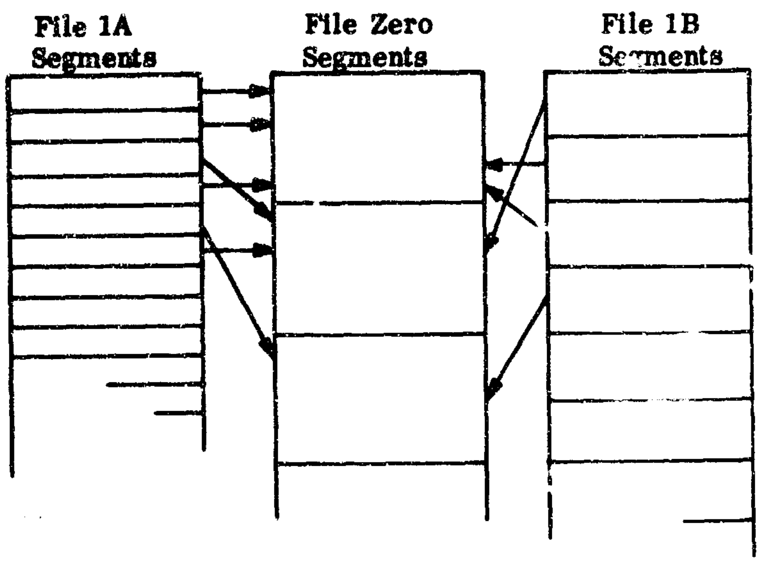
3.1.3.2.3 Indexing. Indexes in GIS can be organized in any of the four ways shown in Figure 3-6. The simple indexes shown in Figures 3-6(a) and (c) will index fields occurring in the Level 00 segment of a file. The compound index shown in Figures 3-6(b) and (d) can index fields occurring in either a Level 00 segment or a Level 01 segment. Either a linear or split physical organization can be employed in a simple or compound index. The index value entries are ordered by data-field value. In the case of a linear compound index, the entries are ordered on both the argument (a Level 00 field value) and sub-argument (Level 01 field value).

3.1.4 Comparison With DM-1

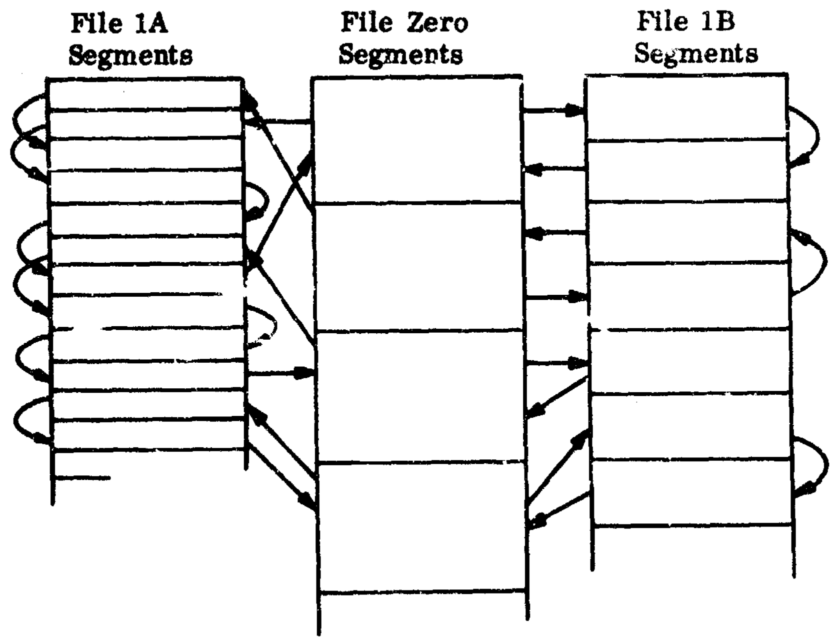
GIS will offer a programming language, a user-oriented query and command language, and a set of basic data management functions such as data definition, update,



a) Linear Record Format

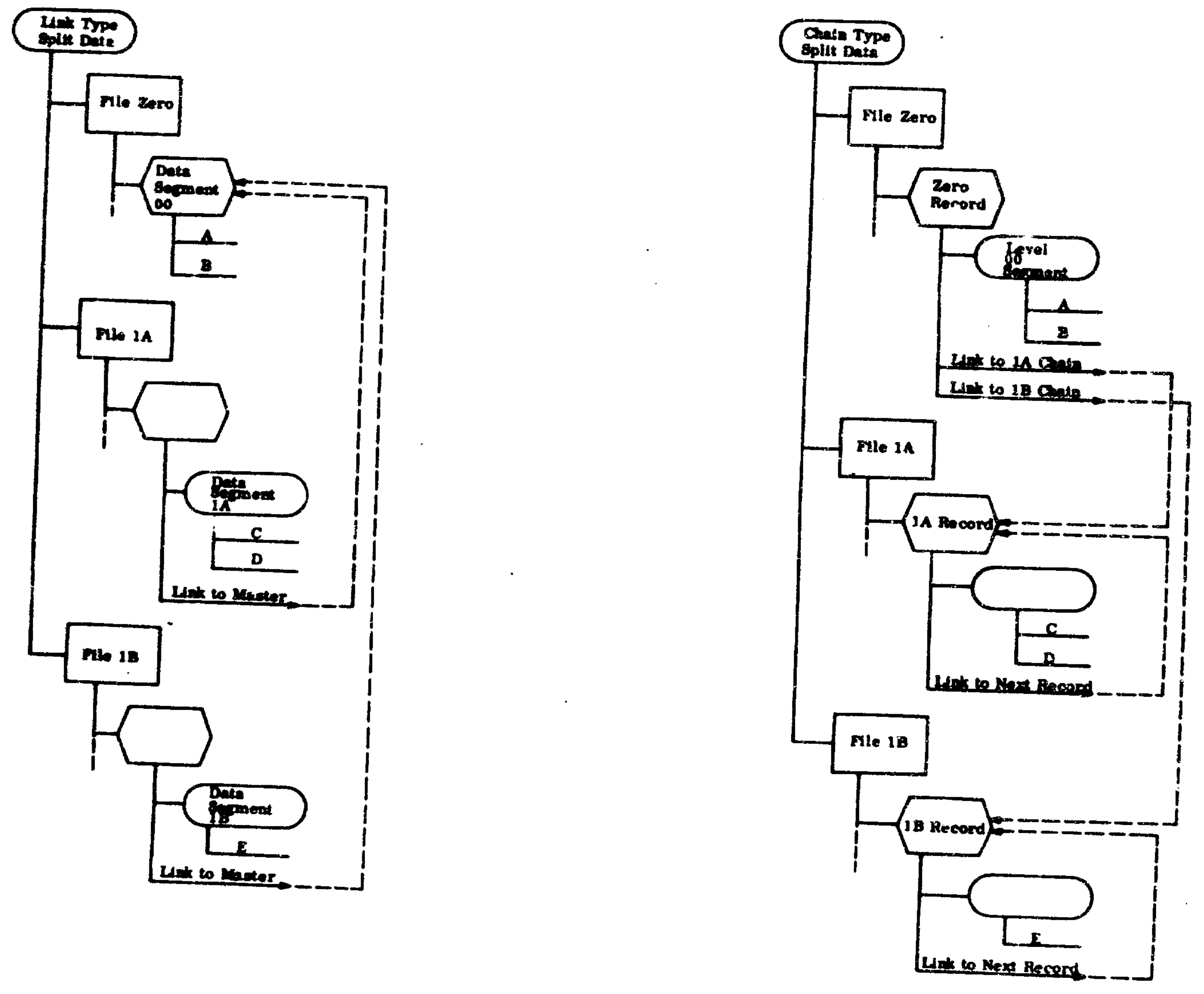


b) Link Type Split Records



c) Chain Type Split Records

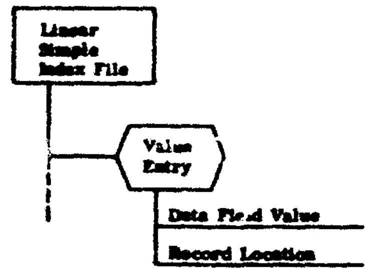
Figure 3-4. Physical Formats



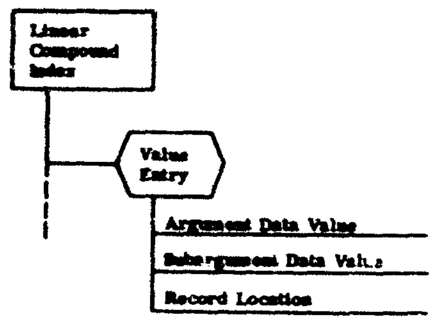
a) Link Type

b) Chain Type

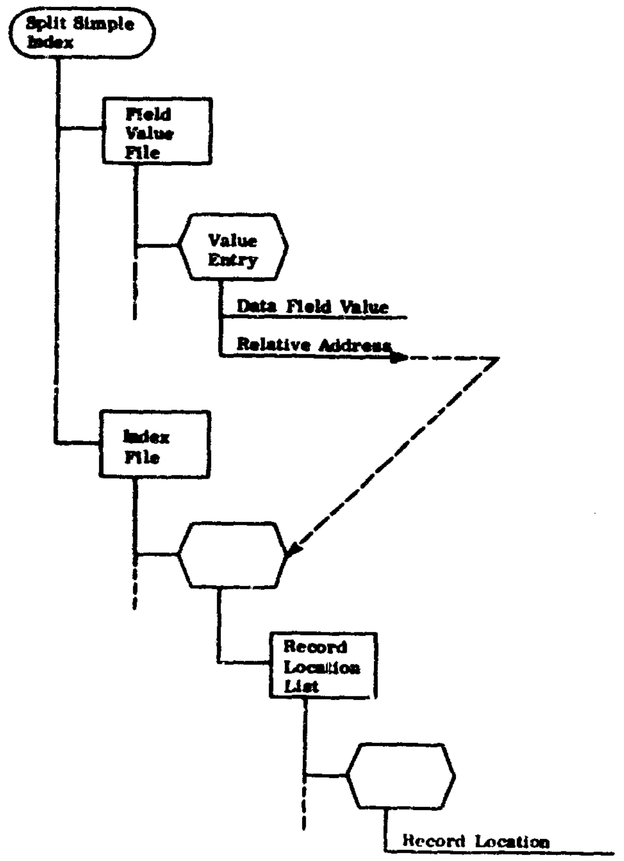
Figure 3-5. Split Records



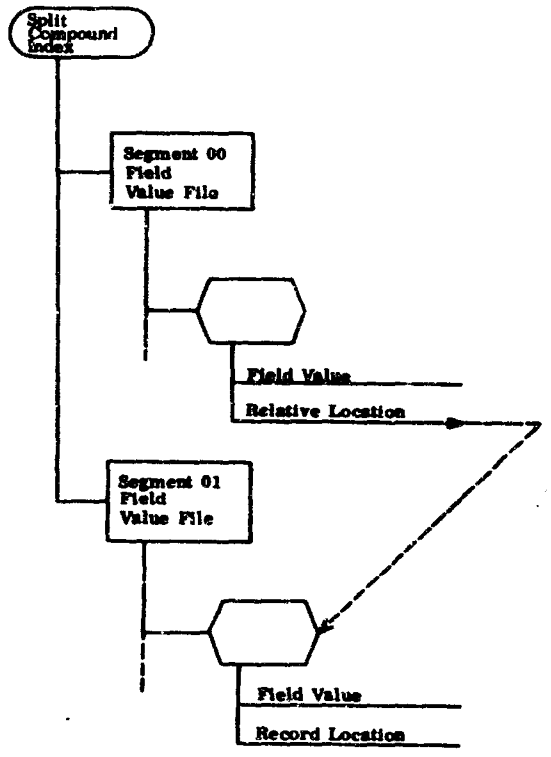
a) Linear Simple Index



b) Linear Compound Index



c) Split Simple Index



d) Split Compound Index

Figure 3-6. Index Types

and query. GIS programs are compiled by the GIS translator and become jobs which run within Operating System/360. Thus, although constrained to a particular operating system, GIS provides the user with a data management system of broad capability. There is a capability of incorporating user-supplied non-GIS tasks in the GIS job. These may be written in other procedural languages such as FORTRAN or COBOL but since they may make no use of GIS services, and since they are bound to parameter and data specifications at compile time, they do not have the advantage of the data independence of GIS programs.

DM-1 differs from GIS in that the operating system functions of job control are integrated with DM-1, whereas no procedural language, as such, is part of DM-1. As a result, the user has a uniform interface with the system for entering the definitions of jobs to be run, and for triggering their execution. The data-access services to tasks running within DM-1 are provided by resident, reentrant system routines that are accessible via calls in the object program which are executed dynamically. With the exception of I/O and the necessity of describing I/O parameters to DM-1, program development proceeds as normal, utilizing the languages appropriate to the problem.

Both GIS and DM-1 have a query language with capability of expressing a logical condition of complete generality. Likewise, both systems have indexes which can be used for efficient retrieval of data which satisfies the conditional expression. DM-1 uses an efficient search strategy which utilizes processing that is largely confined to the index in determining the logical position of data satisfying the condition. In GIS, the indexes use physical, rather than logical, addresses to refer to data, thereby implying more direct access to data but additional maintenance due to physical data movement. In GIS, several index structures are offered, but it is not clear from available documentation whether these are options open to the programmer or whether the choice of index structure is implied by the type of data structure used. It is likewise unclear whether the simple index structure is used efficiently (e.g., logical product searches) to answer complex queries. GIS offers the capability of using masks on the compared values to provide for a "don't-care" capability within a field. DM-1 does not presently have this capability (although it can be added). Also available within the GIS query is the use of several summary operators such as COUNT and AVERAGE. This capability is also potentially available in the DM-1 query but will be implemented initially as analytic tasks (invoked in the job using the query).

In summary, both GIS and DM-1 are high-capability data management systems with some similarity in techniques used, such as data and index structure and query and job languages. Considerable differences exist, however, in the approach taken in the two systems in the manner of interface of the data management system with the programmer and with the operating system. The GIS programmer must use the GIS language as his procedural language and compile an object program which runs as a job within the operating system. The DM-1 programmer uses standard procedural languages such as COBOL. The data services of DM-1 are available to the programmer as resident executive level services that are integrated as an operating system/data management system environment.

3.2 IDS

3.2.1 Objectives

IDS (Integrated Data Store) is a programming system for implementing data processing systems or applications. Through extensions to the COBOL language and compiler, IDS permits the programmer to use mass random-access storage as an extension of memory. Furthermore, IDS is intended to give the programmer an efficient data organization technique, and a language in which to implement data processing applications and systems. Some of the advantages claimed for the IDS programmer (as compared to the COBOL programmer) are as follows:

(1) A reduction in time for the design and programming involved in the implementation of business systems.

(2) A technique for organizing records that is based on their meaning and association with related records in a mass storage environment.

(3) An efficient utilization of mass storage devices.

IDS provides a method of describing complex information structures through the association of data record contents. Once the data is described, the system automatically creates a physical structure which suits the hardware requirements of the mass storage device. The task of organizing data sets for meaningful association is handled by the IDS System. This association is achieved by the use of chains, which provide cross-reference linkages between records of a file.

The COBOL language offers the programmer services in field and sequential record processing. However, it is inadequate when processing records in the random environment of the mass storage. COBOL language statements such as those for read/write operations produce serial rather than random actions. The burden of organizing data records and the responsibility for the logic involved in processing and maintaining these records is placed upon the programmer. IDS extends the range of the COBOL language by incorporating the four powerful IDS instructions -- STORE, RETRIEVE, MODIFY, and DELETE -- as extensions to the COBOL language. These perform the logical functions of record storage, retrieval, modification and deletion. These macro-instructions work in conjunction with, and as a supplement to, the normal COBOL language.

3.2.2 Implementation and Status

The Integrated Data Store is a product of the General Electric Company's corporate research into integrated business systems. Its earliest roots extend back to the system of Report and File Maintenance Generators developed by the General Electric Company at Hanford, Washington. That development work was culminated in the development of 9PAC (later 90PAC), the Report and File Maintenance Generators for the IBM 709. The 9PAC ideas, which were well suited to serial files, were further generalized to handle the capabilities and help solve problems associated with random-access storage.

The Integrated Data Store is presently operational on the GE-200, GE-400, and GE-600 series computers. The GE-200 implementation was first used on a production basis in early 1964. It requires 16K of memory for minimum use. The GE-200 IDS differs from the GE-400 and GE-600 IDS in that it was integrated with the general assembly language for the GE-200 rather than with COBOL. This caused some clumsiness in the use of the language. However, the performance was not affected. IDS itself is implemented in assembly language.

3.2.3. System Description

3.2.3.1 Orientation. IDS is oriented for use by system designers and programmers rather than system users. That is, IDS does not contain a job request language, a job specification language, a query language, or other user-oriented languages. It contains a programmer language made up of the COBOL language plus a series of extensions designed to augment COBOL by introducing a capability of processing linked data structures. A description of the IDS System will be broken down into paragraphs concerning the file structure in IDS, access mechanisms, processing capability, and programmer language description.

3.2.3.2 File Structure. IDS provides the ability to store and retrieve records of any length within a mass memory. Each record type is defined by COBOL record definitions to contain a specified number of fixed-size fields and to participate in a specified number of chains. Different record types have different fields, may participate in potentially different chains, and, therefore, may have different record lengths. The data fields may be any format permissible within COBOL.

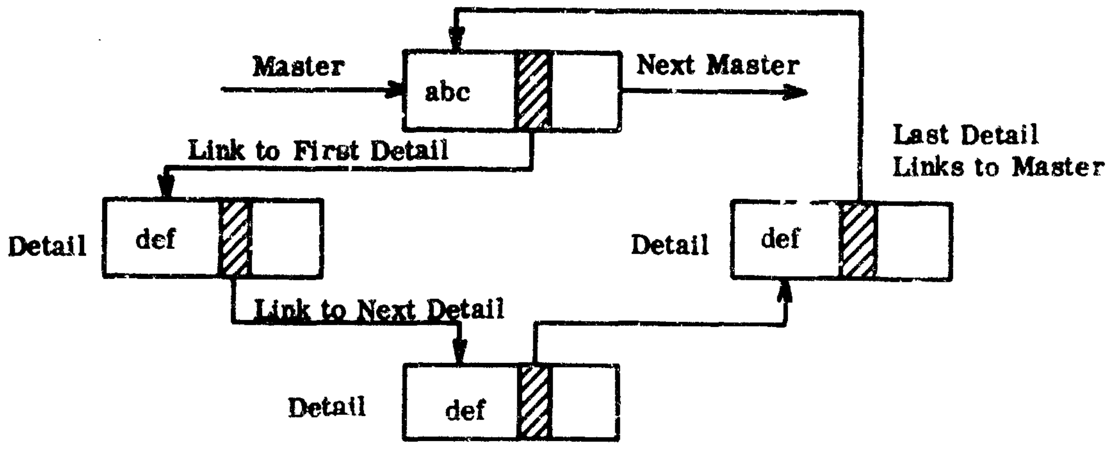
The most characteristic feature of IDS is its file structure. The IDS File structure follows a threaded list format similar to that described by Perlis. (A threaded list is a linked list structure in which the last item on every list is linked back to the parent item that started the list)*. In IDS, each record may be an element in a linked list (since items are in a threaded list structure). A file of records may be subordinated to a master record by linking to the first member of the subordinate file and chaining from that point through each record in the subordinate file through the last one and then back to the master record. This closed-loop chaining is characteristic of all detail record structures in IDS. A diagram of this structure is shown in Figure 3-7(a).

In the notation of a tree diagram, this data structure would be represented as shown in Figure 3-7(b). If stored as a contiguous string, this would be logically equivalent to the embedded structure shown in Figure 3-7(c).

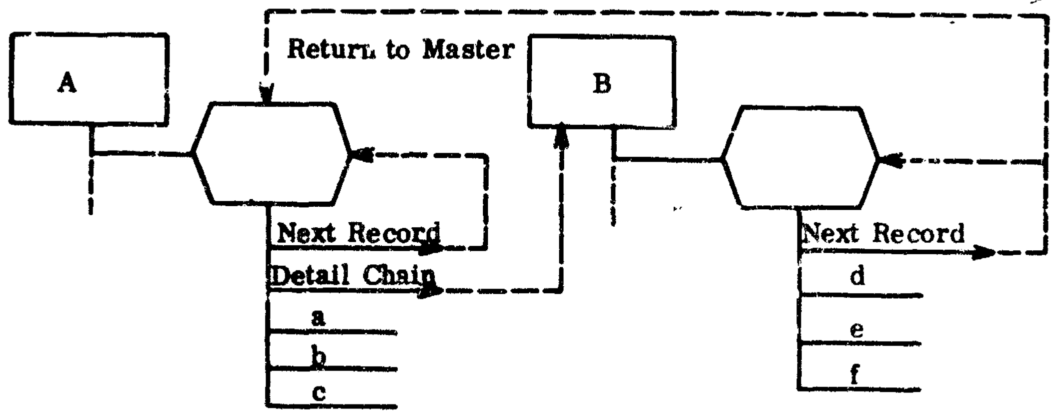
There is no limit to the number of records that may exist in a chain or to the number of detailed chains associated with a single master record in IDS. Likewise, there is no limit to the depth of nesting that is permitted; i.e., a record in a chain that is subordinate to a given record may in turn have other subordinate record chains. A multilevel structure representing a hypothetical data base is given in Figure 3-8. The IDS chaining approach to this structure is shown in Figure 3-9.

The items at a given level in an IDS record are fixed-format, fixed-length records in the COBOL tradition; i.e., the length and format of a specific type of record are fixed by the specifications of the system designer. Records may have any number of data fields, each of which is defined as some number of decimal, alphabetic, or alphanumeric characters. Each record contains an identification area (which will be discussed

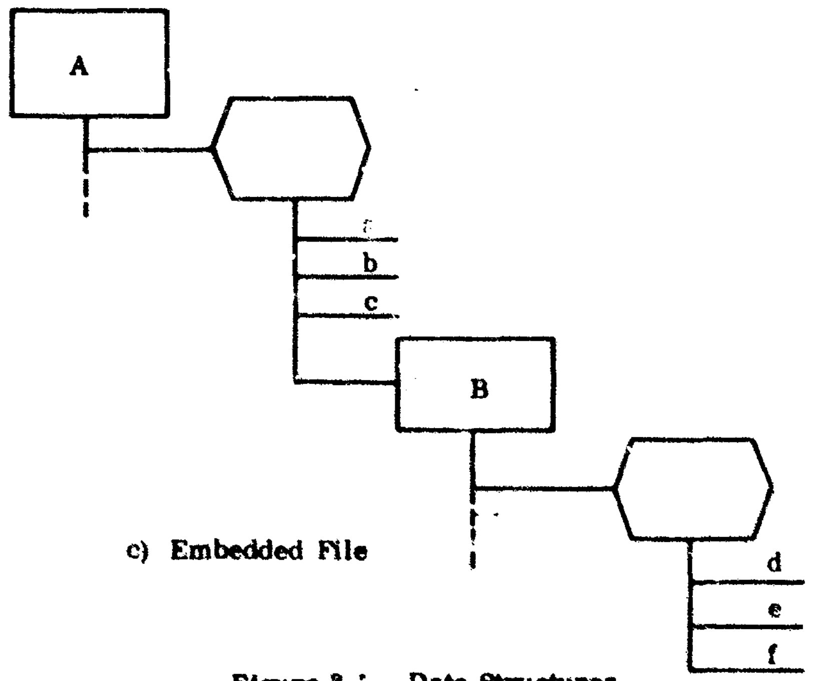
* A.J. Perlis and C. Thornton, "Symbol Manipulation by Threaded Lists," CACM 3.4 (April, 1966), 195-204.



a) IDS Detail Chain Representing an Embedded File in a Master Record



b) Tree Notation for IDS File



c) Embedded File

Figure 3-[illegible]. Data Structures

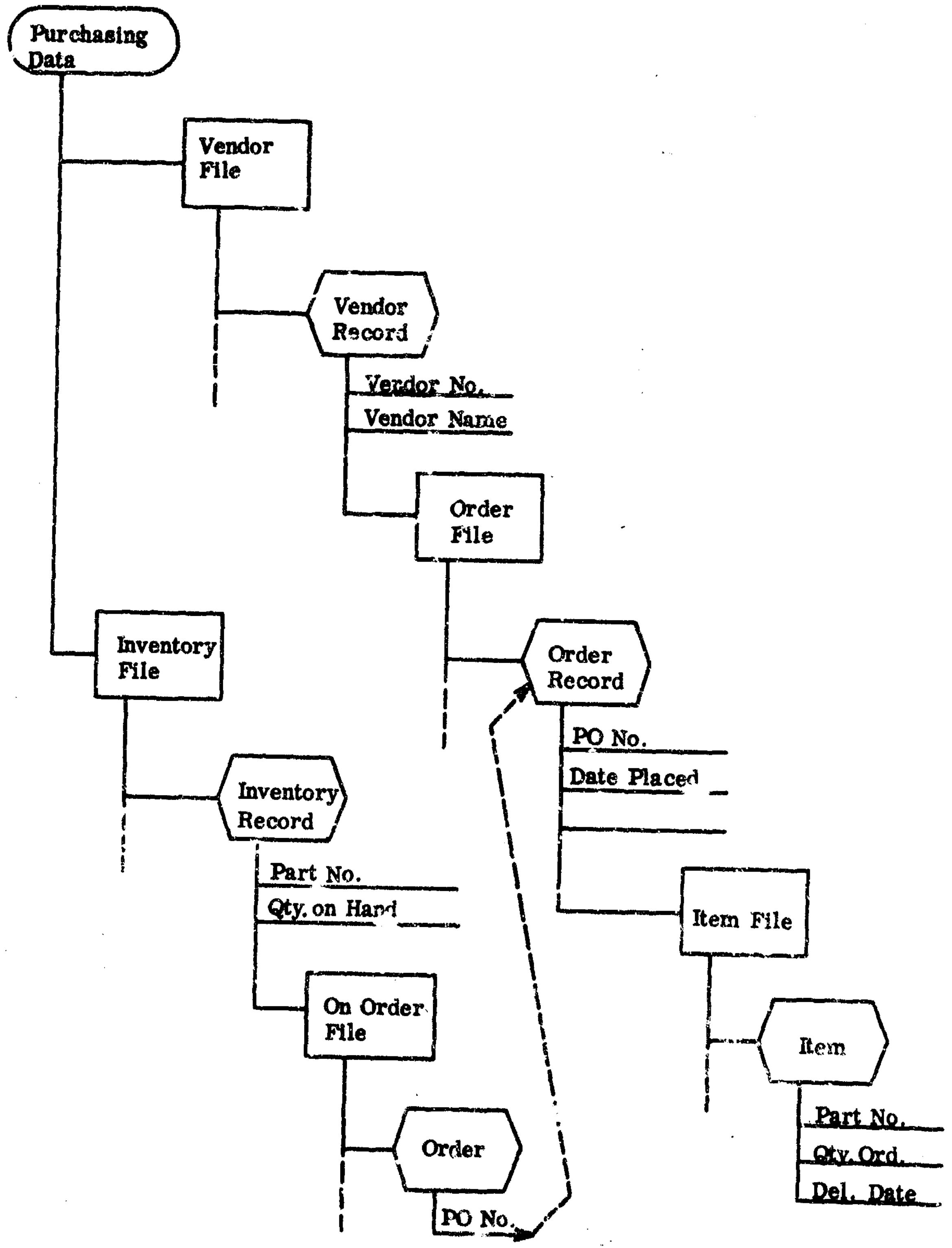


Figure 3-8. Purchasing Data Example

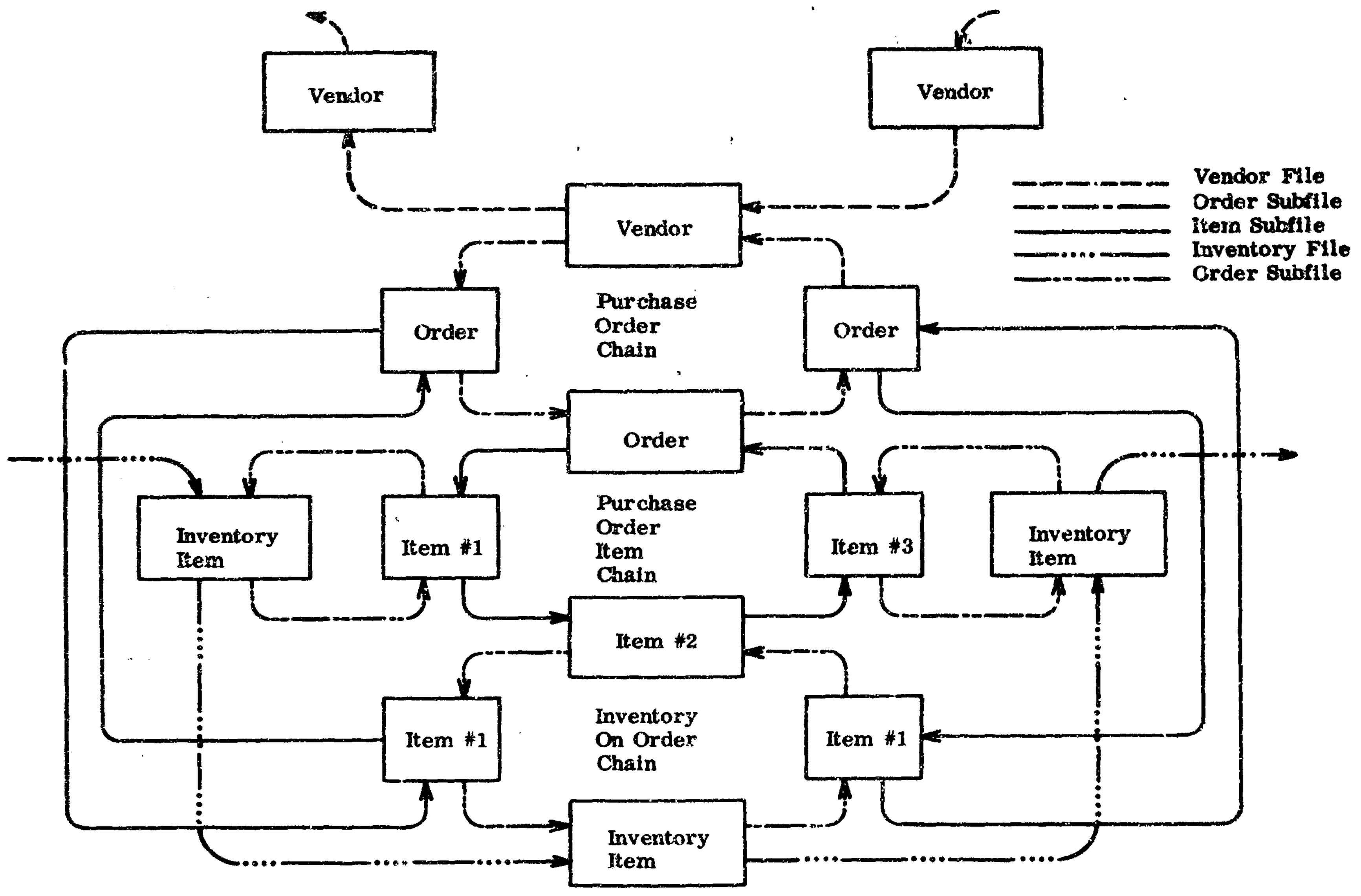
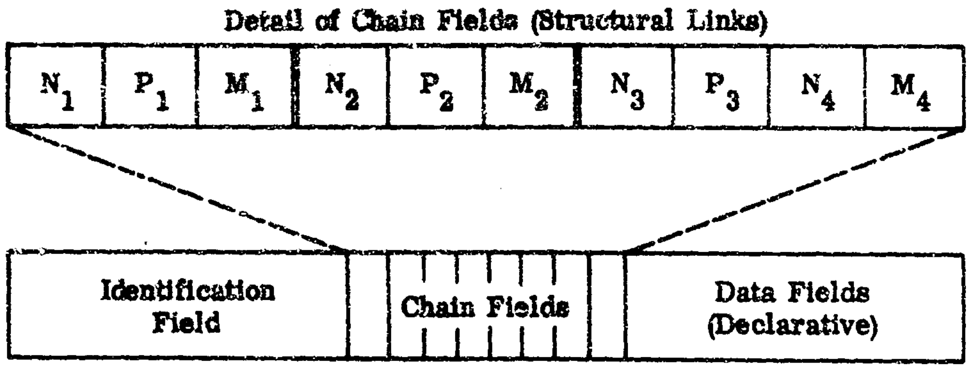


Figure 3-9. IDS Chain for Purchasing Data Example


Detail of Chain Fields (Structural Links)
N1
P1
M1
N2
P2
M2
N3
P3
N4
M4
Identification Field
Chain Fields
Data Fields (Declarative)

N_i = Link to next detail record of chain i

P_i = Link to prior detail record of chain i

M_i = Link to master record of chain i

Figure 3-10. Format of IDS Records

below), a declarative area which can contain information concerning the objects or subject matter of the file, and a third area which contains structural links to other records. These structural links consist of the address of the next record in the chain, the address of a detail, a subrecord to this one, or, optionally, links to prior records in the chain or the master record of the chain. This threaded-list approach permits very flexible data organizations to be represented in the system. Not only is the arbitrary nesting of one file within a record of another permitted, but records may be linked according to arbitrary criteria of similarity. The format of a typical IDS record is shown in Figure 3-10.

The logical structure is mapped into a fixed block size on random-access storage, e.g., disc. A block may contain any combination of record types that are linked into their respective chains. Space is fully utilized by the automatic packing of these records within the block. During processing, complete blocks are read into memory. Therefore, with proper data organization, it is hoped that much of the needed information will be available in a single file access.

Every block begins with a header record. This record contains several control fields used by the system, as follows:

(1) Reference address of the block,

(2) Space available in the block for additional records, and

(3) An I/O control symbol that indicates whether the block has been altered since retrieval.

Records within the block contain an identification field which gives the relative location of the record in the block, the record type, and the record length.

3.2.4 Access Mechanisms

An input-output controller in IDS controls the mass storage device. Its major function is to control the flow of blocks of records in and out of the paged memory in response to commands to store, retrieve, modify, and delete specific data records. To minimize the mass storage seek-and-transfer time, an inventory of data blocks is maintained in memory. These blocks are stored in buffer areas in memory. The number of buffer pages depends on the amount of space available after the IDS subroutines and the problem-solving routines have been loaded.

The greater the number of data blocks stored in core memory, the greater the possibility that the one that is needed next will already be there. To further improve the possibility of finding the desired block in memory, the input-output controller keeps track of the sequence of block utilization and holds the most recently active blocks in memory. Blocks which are infrequently accessed are retired from memory as others are called in. The input-output controller notes which pages have been modified and writes only the modified pages back to mass storage.

Once the input-output controller finds a place for the block in memory, it locates the record called for in the page. The fields from the record will be unpacked into working storage if a MOVE TO WORKING STORAGE command is specified.

3.2.5 Programmer Language

IDS provides the programmer with the ability and requirement to predefine his records, their data fields, and their linkage fields. Once these records and fields have been defined, the programmer is free to operate upon the records without further concern for the details of input or output, the linkage of records, or the protection of the data from erroneous access.

All hardware manipulation and information processing control of the disc storage units is carried out automatically to achieve the four major processing functions:

(1) STORE — stores information and links it into chains,

(2) RETRIEVE — retrieves information from the file,

(3) MODIFY — changes or updates information and relinks chains where necessary, and

(4) DELETE — removes information from the file and relinks chains.

3.2.6 Comparison With DM-1

IDS provides a system building tool to the programmer by giving the COBOL language and compiler the capability to process chained data structures in random-access memory. This COBOL programmer orientation necessarily precludes the use of other procedural languages for IDS programs, but it allows IDS to be uncommitted to a particular operating system, command languages, or data management strategy.

DM-1, on the other hand, is largely independent of any one procedural language. In that sense, it is an operating system which can accept and run object code that is generated by a variety of translators. It provides data-access services dynamically at run time through program calls to resident service routines. Definition of data structures and program parameters are separate processes that are executed by system support functions and which are prerequisite to task execution.

From the definition given in Section II, it can be seen that IDS is not a data management system by itself, since it lacks a user-oriented interface. Rather, it is a set of tools which permits the COBOL programmer to solve a specific data management problem. More generally, by providing random-access-oriented, logical-record processing capabilities which are not specifically avaibable in the COBOL language, IDS permits the programmer to design and implement a specific data management system that is suited to a user's requirements. No query language or other user-oriented command language is built into IDS. By being restricted to chained data structures, however, an essentially sequential search strategy must be used. Also, with chained structures, no index processing is possible, and link addresses are embedded with the data. Powerful techniques for performing logical product searches, such as address list intersection, are not effective with the chaining type of access used in IDS.

DM-1 uses a centralized data structure definition table and physical contiguity of structurally related data (within a block), in addition to logical linkage. This results in compact data sets and efficient utilization of physical storage. Direct physical linkage is used in only one table in DM-1. Another difference between IDS and DM-1 is the variability of items permitted. Fields may be of variable length (including zero) as well as fixed length. Variable length items are stored in variable length space. It is also permitted to define items (of any structure) which need not be accounted for when they are missing (in input data), and no space is dedicated for missing optional items in storage.

3.3 MULTICS

3.3.1 Objectives

MULTICS (MULTiplex Information and Computing Service) is an operating system for the GE 645 computer system being developed by three organizations: The Bell Telephone Laboratories, the General Electric Company, and Project MAC at MIT. The system will be the standard operating system for that computer — a multiprocessor for simultaneous processing of real-time and batch jobs.

MULTICS is being developed as a monitor which will accept input from a large number of remote consoles as well as control read-time applications, batch processing, and jobs which are mixtures of batch processing and console requests. An interesting feature of MULTICS is that it is designed specifically with low overhead costs in mind. For example, an important goal is to operate effectively during overload. Scheduling techniques have been specifically designed for this purpose.

The designers of MULTICS view the GE 645 computer center as an open-shop computer "utility" which must provide service on demand and without advance notice. There may be as many as 1000 remote typewriter consoles subscribing to the MULTICS facility. Depending on the program demands of the users, as many as 100 may be active at one time.

The MULTICS designers expect that, although an initial capability is expected to be operational in 1966, the design of MULTICS will continue to evolve. Indeed, the basic design philosophy is to permit and encourage this evolution in capability.

3.3.2 Implementation and Status

MULTICS is now in the preliminary stages of implementation, and detailed design information has not been released. Available documentation* is concerned with outlining goals and the techniques to accomplish these goals. Specific procedures for use of the system have not been established. The language in which MULTICS is being programmed is PL/I.

The minimum hardware configuration with which 645 MULTICS can run is one 645 CPU, 64K of core memory, one high-speed drum or one disc unit, four tape units, and eight typewriter consoles. However, MULTICS will not run efficiently on this minimum configuration, and would normally be operated thus only when a substantial part of a larger configuration was unavailable for some reason.

A small but useful hardware complement would be 2 CPU units, 128K of core, 4 million words of high speed drum, 16 million words of disc, 8 tapes, 2 card readers, 2 line printers, 1 card punch, and 30 consoles.

The initial implementation of 645 MULTICS software is designed to support a maximum configuration of up to 8 CPU's, up to 16 million words of core, up to 2 high speed drums, up to 300 million words of disc and disc-line devices, up to 32 tapes, up to 8 card readers, 8 punches, 16 printers, and up to 1000 or more typewriter consoles. It will not, of course, operate efficiently (or in some cases at all) with an arbitrary and unbalanced mixture of these. For instance, 645 MULTICS would not run well with 6 CPU's and 128K words of core.

3.3.3 User Languages

Specifications for user languages are not present in any available documentation. It might be reasonable to assume, however, that a Job Specification Language is contemplated for the MULTICS system.**

* The sole source of information for this report is the set of six papers presented at the 1965 FJCC and referenced in the bibliography. Design details of MULTICS have not been released as yet.

** This is conjecture, but, since the GE-645 is to operate as a time-shared system, it seems reasonable.

The lack of a query language and item awareness at the field level keep MULTICS from being considered a data management system.

3.3.4 Job Processing Strategy

Since MULTICS is meant to be the operating system of a computing utility that serves many on-line users, a basic aspect of its character is discernible from its job processing strategy. The designers claim that a computing facility that is almost never overloaded is spending too much money for computing hardware and that the system state will oscillate between/overload and underload. They further observe that the performance of the operating system must be judged during the periods of overload, and that denial of service to new users during these periods is better than degrading service over the entire population. The following somewhat contradictory statements are made: the system must provide service on demand without advance notice and without batching or prescheduling; the scheduler should get information concerning the urgency of jobs from human beings, and the scheduler should not have any built-in assumptions that console jobs are either more or less urgent than absentee jobs, or that short runs are either more or less urgent than long runs. They do not resolve the question of how it is possible to get an indication of urgency from a console that has been locked by the executive program and denied service because of an overload.

3.3.5 Processing Capability

In the sense of a general data management system, MULTICS has very little processing capability since it is quite broad in scope of application and deals with a data structure whose elements are files, records, and blocks. However, it is designed to be an operating system with more than the standard capability for multiuser control, efficient paging, and data-access protection.

MULTICS does not include a query system or other user-oriented data management function. However, since it is a general-purpose operating system, MULTICS will support jobs and provide all data management services, and, presumably, will possess the usual complement of service and utility functions.

3.3.6 File Structure

MULTICS data management functions are concerned mainly with the most general types of data set manipulation. This [illegible] most of the functions deal with the data base only as a set of files. There is little provision (published) for manipulation of the contents of the files, except at the physical level (i.e., blocks). One exception to this very general statement is the fact that the user has at his disposal the ability to create and manipulate a directory or directories of files. Since a directory is basically a file and allows the manipulation of named data sets (for example, there are data-linking operators), some intermediate level data structuring facility is offered.

These comments are consistent with the notion that a file is formatless at the system level. In general, this contributes to the fact that the user can view his storage as being infinite and homogeneous. That is, the file designer need not worry about individual formats (fields or records) being consistent with a particular disc, drum, or other storage-medium addressing scheme.

The MULTICS system, as described, offers the facilities to link nodes which are not connected a priori in the basic structure, and to name paths through the tree, as well as naming the nodes. There is also an operator which allows the user to link paths. These facilities are most useful in the MULTICS system if the user wishes to write his own search procedure. A generalized search is offered, but it may be bypassed in favor of one which is individually tailored to a particular file.

A tree hierarchy of directories exists. The directories consist of a list of entries, each of which is a pointer to either a file, another directory, or another entry in the same or another directory. A master directory is at the root of the tree. Descriptive information concerning each entry in the directory is contained in the directory. An access mode is associated with each entry. This assures privacy of files. A tree search of directory branches is used to find a file. Records of the file are then accessed sequentially, with the file being viewed as a linear array with a single subscript. The user may himself reference an element of a file by specifying the symbolic name of the file and its linear index (subscript value). No data structuring capability within the record (fields, sub-files, etc.) is present in MULTICS. Data structuring is completely defined by the user within his program.

Facilities are included for the on-line maintenance of the directory by the programmer/user. Files may be maintained either on-line or off-line.

No output formatting is specified in the available documentation.

The user of MULTICS can assume an essentially unlimited amount of single-level, on-line storage when designing a system. Since the features of most operating systems are included in MULTICS, the user also has at his disposal other items such as dynamic storage allocation and the like.*

It is important to recognize, however, that MULTICS per se is not a data management system as the term is used in this report. The currently described MULTICS system is designed primarily as an operating system and, therefore, lacks many of the features normally associated with a data management system (e.g., it lacks a query language). Furthermore, the notion of file structure appears to be more oriented toward systems operation and the control of program segments, rather than toward data files. However, the file management aspects of the system are such that they warrant analysis in this context.

The designers of the MULTICS system have paid particular attention to the problems of ensuring privacy to a file or a set of files. In general, the MULTICS system will be designed so that permission to access a file will rest logically on the branches (or branch) which point(s) to the file. There are five attributes to a set of permissions which control access to a file (TRAP, READ, EXECUTE, WRITE, and APPEND). These attributes are essentially boolean flags which may be on or off for a particular user. TRAP is said to be logically powerful enough to subsume the other four attributes, which are called usage attributes.

3.3.7 Comparison With DM-1

MULTICS is an operating system for the GE-645 computer which will be able to support a broad spectrum of procedural languages and programming systems for data processing and scientific applications in either user on-line or batch mode. The logical data item services in MULTICS are limited to the file and record level. At those levels, however, complete multilevel directory access can be constructed, with access-rights safeguarded for reading and writing.

* It is important to remember that the user is likely to be the programmer, and that jobs are likely to mean the same as programs.

To a large extent, the data access services of MULTICS complement, rather than overlap, those of DM-1. DM-1 furnishes logical item access services at all levels of logical data structure, but relies on a separate input-output system for physical device I/O and storage allocation. Access rights protection exists at all data levels in DM-1, and specific access rights may be contingent on the value of specified data items.

A basic objective of DM-1 (as well as MULTICS) is to achieve an optimal response to non-planned service requests. The basic strategy in DM-1 which, it is felt, can enable it to achieve this objective is the use of a stored user list and job description list. All job requests will be requests from recognized users to run predefined jobs. (The acts of adding new users or jobs to the lists are themselves predefined system support jobs.) Each user has a known priority and rights for data access, and each job has an estimated running time. The intent in DM-1 is to read all job requests within a reasonable response time and then to schedule job execution on the basis of the known urgency (a function of priority, deadline, and running time estimate) of all jobs in the curre. mix.

SECTION IV. BIBLIOGRAPHY

4.1 GIS

Generalized Information System Application Description, IBM, E20-0179-0 (1965).

GIS Briefing Outline, IBM, GIS Field Support Group (January, 1966)

4.2 IDS

Introduction to Integrated Data Store, GE Computer Department, CPB-1048 (April, 1965).

Bachman, C.W. and Williams, S. B. "A General-Purpose Programming System for Random-Access Memories," FJCC (1964), 411-422.

Bachman, C. W. "Software for Random Access Processing." Datamation (April, 1965).

Proceedings of the Second Symposium on Computer-Centered Data Base Systems, SDC TM - 2624/100/00 (1 December 1965), 3-231 to 3-275.

4.3 MULTICS

Corbato, F. J. and Vyssotsky, V. A. "Introduction and Overview of the MULTICS System," AFIPS 27, (Proc. FJCC, 1965), 185-196.

Glaser, E. L., Couleur, J. F., and Oliver, G. A. "System Design of a Computer for Time-Sharing Applications," op. cit., 197-202.

Vyssotsky, V. A., Corbato, E. J., and Graham, R. M. "Structure of the MULTICS Supervisor," op. cit., 203-213.

Daley, R. C. and Neumann, P. G. "A General-Purpose File System for Secondary Storage," op. cit., 213-229.

Ossanna, J. F., Mikus, L. E., and Dunten, S. D. "Communications and Input-Output Switching in a Multiplex Computing System," op. cit., 231-241.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Auerbach Corporation Philadelphia 3, Pennsylvania 19107	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE
Reliability Central Automatic Data Processing Subsystem

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Final Report

5. AUTHOR(S) (Last name, first name, initial)
Dr. J. Sable, W. Crowley, M. Rosenthal, S. Forst, P. Harper

6. REPORT DATE August 1966	7a. TOTAL NO. OF PAGES 780	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. AF 30(602)-3820	9a. ORIGINATOR'S REPORT NUMBER(S) 1280-TR	
b. PROJECT NO. 5519		
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.	RADC-TR-66-474 (3 Vols)	

10. AVAILABILITY/LIMITATION NOTICES
This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of RADC (EMLI), GAFB, NY 13440.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Rome Air Development Center (EMIID) Griffiss Air Force Base, New York 13440

13. ABSTRACT

This is a three-volume final report produced for the Rome Air Development Center (RADC) under Contract AF 30(602)-3820. Volumes I and II are the Design Specification Report for the Automatic Data Processing Subsystem (ADPS) of Reliability Central, known as Data Manager-1 (DM-1). Volume III is a survey of major, computer-oriented on-line information and fact retrieval systems.

The system design specification will be used for the implementation of the computer programs required to operate the RADC Reliability Central. The work reported in these volumes is an extension and detailing of the functional system design developed by Auerbach Corp. under Contract AF 30(602)-3433 and reported in RADC-TR-65-189, Design of Reliability Central Data Management Subsystem, July 1965. The DM-1 design provides for the incorporation of the reliability data collected by the Illinois Institute of Technology Research Institute (IITRI) under Contract AF 30(602)-3621 with Auerbach Corp. as subcontractor.

DD FORM 1 JAN 64 1473

UNCLASSIFIED
Security Classification

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
File Structures						
Programming Languages						
Data Processing						
Storage and Retrieval						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through ________________."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through ________________."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through ________________."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

UNCLASSIFIED

Security Classification